Project: Bank Marketing (Campaign)

Week 12 deliverables

Group name: Fleifel-solo

Name: Rania Tarek Fleifel

Email: raniatarekfleifel@gmail.com

Country: Egypt

College: Cairo university Faculty of engineering

Specialization: Data science

Internship Batch: LISUM13: 30

Submission date: 20th December 2022

Github repo: https://github.com/RaniaFleifel/Data-glacier-internship/tree/main/Data%20science%20project_Bank%20Marketing%20Campaign/week12

Choice of metric:

There are various metrics that can be applied to evaluate the performance of predictive models. Among these metrics are accuracy, precision, recall, f1, log-loss, roc-auc curve and R2. The prevailing logic in picking the metrics that suits the project's problem description and optimize the models accordingly.

In this project, the ultimate goal is to have the marketing team target costumers that are likely to invest in the deposit term product. Since the point is efficient management of the team's resources, we don't want the team to waste time on costumers who wouldn't invest. In the context of the confusion matrix, this means that the main concern is having a lot of False positives. Precision is our metric of choice since it measures the number of true positives divided by number of true and false positives.

Predictive models proposed:

1) **Logistic regression**

- Reason of choice: This classifier is considered the benchmark of binomial problems. Sigmoid function maps the predicted values to probabilities that best distinguish the classes.
- Scaling: sigmoid function requires the data to be between 0 and 1. We take this opportunity to test the transformation done on the numeric data and how well it generalizes when used with MinMax Scaler. Usually two scalers are used with LR, minmax scaler and stdscaler. It is advised to use stdscaler if you have gaussian-like distribution in the features which is not the case.
- Hypothesis (what can this model handle): LR is sensitive to outliers and skewness because of the scaling. If not handled beforehand, the outliers and skewness can cause the significant range of information to be squished down. It uses regularization that normalizes across rows (per single entry). This is also why it inherently requires all features to be scaled to the [0,1] range so the stronger features don't over-ride the smaller features.
- Hyperparameters tuning (grid search)
    - To account for the imbalanced dataset, we set class_weight: 'balanced' that adjusts class's weight according to class's frequency

in the input data. This is equivalent to using SMOTE to resample the classes.

  - In the regularization penalty field, we test two options “none” and “L2”. Regularization generally is aimed at preventing overfitting. L2 shrinks all features’ coefficients evenly.
  - In the multi_class hyperparameter: we try both ovr and multinomial. Ovr is used as a one-vs-all method of approaching features while multinomial considers that some features have some dependency on one another
  - As for the solvers used, we attempt two solvers sag and newton-cg that can handle our choices of multi_class and regularizations.

- Best model

```
best parameters are {'lr__class_weight': 'balanced', 'lr__multi_class': 'ovr', 'lr__penalty': 'l2', 'lr__solver': 'sag'}
```

It’s expected that the multi_nomial mode would surpass ovr because we’ve previously discussed the underlying relation -that exists- between features. Also the prevalence of no penalty suggests that our features would generalize well and not overfit.

- Its statistics

Classifier	Log loss	Accuracy	Precision	Recall	F1	Roc auc	R2	time
lr	0.514867	0.828688	0.358833	0.629344	0.457063	0.741914	-0.688654	0.078

The model does a bad job identifying positives (precision).

** We suspect that the reason might be the ambiguous values “unknown” or the forwardfilled values in eurobor3m. So we attempt to exclude all samples that had ambiguous values and rerun the grid.

```
23436 samples from 36164 training samples have no-missing data
best parameters are {'lr__class_weight': 'balanced', 'lr__multi_class': 'multinomial', 'lr__penalty': 'none', 'lr__solver': 'ne
wton-cg'}
```

However, the performance excluding the entries with ambiguous values takes almost the same time to converge as using the fulldataset , but performs worse in terms of precision. As seen above, taking out one-third of the training dataset renders the learning slower.

Classifier	Log loss	Accuracy	Precision	Recall	F1	Roc auc	R2	time
lr	0.514867	0.828688	0.358833	0.629344	0.457063	0.741914	-0.688654	0.078000
lr-missingvalues	0.498598	0.812763	0.329881	0.614865	0.429390	0.726618	-0.845636	0.070004

2) Random Forest (Base model)

- Reason of choice: the ensemble of decision trees seems like a perfect match for this problem; it handles a number of features per tree with replacement and then uses majority rules to make final decisions. This resembles how we approached features for analysis; dividing and conquering.
- Scaling: Random forest classifiers do not need scaling of data. However, it might be a good idea to use the log-scaled version of the numeric values to avoid having the trees only correctly predict on trees with high-end values.
- Hypothesis (what can this model handle): RF is supposed to be immune to missing/ambiguous values since it handles a small percentage of features per tree then votes among all learners. Its no need for scaling removes another step of the process that required careful consideration
- Hyperparameter tuning (grid search)
    - We attempt max_depth of ½ and 2/3 of length of the the training sample
    - The number of estimators: 150
    - The feature selection criteria: entropy, gini
      Entropy is the measure of disorder/impurity per node. It ranges from 0 to 1, where 0 describes nodes with only one class such as leaf nodes.
      Gini on the other hand measures how probable it is to misclassify an instance chosen randomly. The lower the gini index, the better the likelihood of misclassification.
    - Class_weight: balanced, balanced_subsample
      The difference between both is that the prior one uses samples frequency to determine class weight while the latter one does the same but for each individual tree.
    - Max_samples: 0.2,0.4

This determines how many samples to use per tree

- Oob_score: True
  This validation method uses out-of-bag samples to reach a gerenarlized score

- Best model

```
best parameters are {'rf__class_weight': 'balanced_subsample', 'rf__criterion': 'gini', 'rf__max_depth': 36164, 'rf__max_sample
s': 0.2, 'rf__n_estimators': 150, 'rf__oob_score': True}
```

The best precision is achieved when the class weight is determined per tree and depth of half the training sample size.

- Its statistics

Classifier	Log loss	Accuracy	Precision	Recall	F1	Roc auc	R2	time
lr	0.514867	0.828688	0.358833	0.629344	0.457063	0.741914	-0.688654	0.078000
lr-missingvalues	0.498598	0.812763	0.329881	0.614865	0.429390	0.726618	-0.845636	0.070004
rf	0.319384	0.898031	0.654054	0.233591	0.344239	0.608801	-0.005125	0.573804

There's considerable improvement in precision compared to LR, almost doubled.

** random forest is not sensitive to ambiguous/missing data like LR. To validate this, we attempt the search grid on training samples that have no ambiguous values in any feature, we actually find the tree performs worse that the case with the whole dataset. We also notice that unlike the training on the whole dataset, in this case weighing classes on the entire training sample yields better precision

```
best parameters are {'rf__class_weight': 'balanced', 'rf__criterion': 'gini', 'rf__max_depth': 11718.0, 'rf__max_samples': 0.2,
'rf__n_estimators': 150, 'rf__oob_score': True}
```

Classifier	Log loss	Accuracy	Precision	Recall	F1	Roc auc	R2	time
rf_missingvalues	0.336615	0.898142	0.641975	0.250965	0.360861	0.616427	-0.004035	0.604405

3) Balanced bagging classifier (RF base)

- Reason of choice: Similar to rf, this classifier is a bagging method that uses several base classifiers of the same type (homogenous) to reach a generalized decision. The difference is that this classifier includes an extra step that resamples the training data to combate imbalanced datasets.

In other words, this classifier is resampling + RF (base)

- Scaling: similar to RF, this classifier doesn't need scaling of features, the log_scale of features suffice
- Hypothesis (what can the model handle well) it's expected that with this extra step, the precision would improve and precision decrease a bit since we're adding an extra balancing step in favor of the minority class.
- Hyperparameter tuning (grid search)
    - We attempt different sampling strategies: majority, not_majority, all The majority sampling undersamples the class that overrides in the statistics (majority) until it's balanced with the minority class, minority (not_majority) sampling oversamples minority class to match majority and all resamples all training samples to reach the required balance.
    - We use the best model reached using RF without research grid.
- Best model

```
best parameters are {'bag__sampling_strategy': 'not majority'}
```

As expected, the model yields the best precision when it oversamples the minority class not the other way around. The other way around is recommended only when the dataset is huge in the first place. But in our case, so much information is dropped in case of undersampling the majority class.

- Its statistics

Classifier	Log loss	Accuracy	Precision	Recall	F1	Roc auc	R2	time
lr	0.514867	0.828688	0.358833	0.629344	0.457063	0.741914	-0.688654	0.078000
lr-missingvalues	0.498598	0.812763	0.329881	0.614865	0.429390	0.726618	-0.845636	0.070004
rf	0.319384	0.898031	0.654054	0.233591	0.344239	0.608801	-0.005125	0.573804
bag_rf	0.277636	0.899027	0.666667	0.237452	0.350178	0.611044	0.004686	14.695433

As expected, there's an improvement in the precision when the extra balancing step is added to the model.

4) XGBoost classifier

- Reason of choice: At the heart of it, it is still a group of decision trees, but what makes it superior in terms of performance is that it uses gradient descent to fix errors of trees that joined the model as it goes and keeps adding trees until no further improvements can be made.
- Scaling: Doesn't require scaling similar to all previously discussed tree-based models
- Hypothesis (what can the model handle well): we make use of the hyperparameter scale_pos_weight that pays more attention to misclassifications of the minority class. This is specifically helpful in cases of severe imbalance.
- Hyperparameter tuning (grid search):
    - scale_pos_weight:

$$\frac{2}{estimate}, \frac{3}{estimate}, \frac{5}{estimate}, \sqrt[5]{estimate}, \sqrt[3]{estimate}, \sqrt[2]{estimate}, estimate$$

we set the scale_pos_weight relative to the ratio between the positive to negative class in the training set (estimate), encouraging the model to over-correct the positive class. We need to make sure we don't overdo it, or else the negative class will be hugely misclassified in the process. Which is why we attempt to use the roots of the estimate, since usually the pos_weight=1/estimate in case of severely imbalanced dataset (1:100) for example.

- Estimate=total majority examples/total minority example

Best model

```
Counter({0: 31913, 1: 4251})

Estimate: 7.507
best parameters are {'xgb__scale_pos_weight': 0.2664118071005546}
```

Since our most important metric of performance is precision, it makes sense that the weight that performs best is the one that gives most advantage to the minority class.

Its statistics

Classifier	Log loss	Accuracy	Precision	Recall	F1	Roc auc	R2	time
lr	0.514867	0.828688	0.358833	0.629344	0.457063	0.741914	-0.688654	0.078000
lr-missingvalues	0.498598	0.812763	0.329881	0.614865	0.429390	0.726618	-0.845636	0.070004
rf	0.319384	0.898031	0.654054	0.233591	0.344239	0.608801	-0.005125	0.573804
bag_rf	0.277636	0.899027	0.666667	0.237452	0.350178	0.611044	0.004686	14.695433
rf_missingvalues	0.336615	0.898142	0.641975	0.250965	0.360861	0.616427	-0.004035	0.604405
xgb	0.318658	0.894382	0.728814	0.124517	0.212696	0.559261	-0.041100	0.124800

As expected, this choice of weight does improve the detection of the minority class, but it also confuses the majority class quite a lot. I believe this trade-off is still sensible in our problem. This result means that if the marketing team focus on the positively-predicted costumers, 72% of them will go-for the deposit product.

How does the statistics look like with “duration” taken in consideration?

It’s important to mention that the previous results are obtained while excluding “duration” feature. This feature is problematic from a logical perspective since the duration of the call is registered After the attempt to convince the costumer is already concluded. So the outcome of the conversation is already known as well.

Classifier	Log loss	Accuracy	Precision	Recall	F1	Roc auc	R2	time
lr	0.358746	0.852577	0.428845	0.863900	0.573167	0.857506	-0.453180	0.062400
lr-missingvalues	0.312609	0.881000	0.486339	0.687259	0.569600	0.796665	-0.173009	0.093005
rf	0.187899	0.909202	0.690941	0.375483	0.486554	0.676874	0.104981	0.444202
bag_rf	0.183512	0.910529	0.694017	0.391892	0.500925	0.684767	0.118063	12.503869
rf_missingvalues	0.260553	0.906990	0.661157	0.386100	0.487508	0.680247	0.083178	0.495002
xgb	0.251024	0.897700	0.740260	0.165058	0.269929	0.578782	-0.008395	0.151802

As expected, the precision improves for all models, especially for LR, when duration is taken in consideration, though adding it to the analysis is not

logistically sound. The improvement in recall is even bigger, this is because there's an implied definition that precision is used to analyze how well a model will predict a future event, while recall is used to evaluate how well a previous event was predicted. Because of the previously discussed thought that duration is known only when contact with a costumer is in the past, this agrees with improving recall.